A Real-Time Object Recognition System on Cell Broadband Engine

Hiroki Sugano[1] and Ryusuke Miyamoto[2]

[1] Dept. of Communications and Computer Engineering, Kyoto University, Yoshida-hon-machi, Sakyo, Kyoto, 606-8501, Japan
hiroki@easter.kuee.kyoto-u.ac.jp

[2] Dept. of Information Systems, Nara Institute of Science and Technology, 8916-5, Takayama-cho, Ikoma, Nara, 630-0192, Japan
miya@is.naist.jp

Abstract. Accurate object recognition based on image processing is required in embedded applications, where real-time processing is expected to incorporate accurate recognition. To achieve accurate real-time object recognition, an accurate recognition algorithm that can be quickened by parallel implementation and a processing system that can execute such algorithms in real-time are necessary. In this paper, we implemented an accurate recognition scheme in parallel that consists of boosting-based detection and histogram-based tracking on a Cell Broadband Engine (Cell), one of the latest high performance embedded processors. We show that the Cell can achieve real-time object recognition on QVGA video at 22 fps with three targets and 18 fps with eight targets . Furthermore, we constructed a real-time object recognition system using SONY® Playstation 3, one of the most widely used Cell platforms, and demonstrated face recognition with it.

Keywords: Object recognition, Cell Broadband Engine, Real-time processing, Parallel implementation.

1 Introduction

Currently we must realize an object recognition system based on image processing for embedded applications, such as automotive applications, surveillance, and robotics. In these applications, highly accurate recognition must be achieved with real-time processing under limited system resources. For such achievement, both a highly accurate recognition algorithm suitable for parallel processing and a real-time processing system suitable for image recognition must be developed.

Generally, object recognition based on image processing is achieved by combining object detection and tracking [1]. For example, a neural network [2], a support vector machine [3,4], and boosting [5] are adopted in the detection phase for pedestrian recognition, one application of object recognition. In some cases, candidate extraction based on segmentation is also adopted to enhance the detection performance [6]. In the tracking phase, recently particle filter-based schemes are widely used [7,8], although Kalman filter-based schemes used to be popular.

On the other hand, some works toward real-time processing of object recognition on embedded systems exist. Some aim for rapid object detection by a specialized processor [9], and others propose real-time stereo that sometimes aids object detection [10]. In such works, Field Programmable Gate Array (FPGA), which is programmable hardware, Application Specific Integrated Circuit (ASIC), which is hardware designed for a specific application, and high performance Digital Signal Processor (DSP) are adopted. However, a highly accurate real-time object recognition system has not been developed yet.

In this paper, we propose a real-time object recognition system that achieves highly accurate recognition. In our proposed system, an object recognition algorithm based on the scheme proposed in [11] is adopted. In this recognition scheme, boosting-based detection and color histogram-based tracking with a particle filter are used for the detection and tracking phases, respectively. Because both have massive parallelism, parallel implementation is expected to improve processing speed. For a processing device, we adopt Cell Broadband Engine (CBE), one of the latest high performance embedded processors for general purpose use, which has a novel memory management system to achieve efficient computation with parallel execution units. By utilizing the computational power of CBE suitable for image recognition, we realize a highly accurate real-time object recognition system.

The rest of this paper is organized as follows. Section 2 describes boosting-based detection and particle filter-based tracking adopted in the proposed system. In Section 3, CBE architecture is summarized and parallel programming on CBE is introduced. Section 4 explains parallel implementation of detection and tracking. In Section 5, a real-time object recognition system on SONY® Playstation 3, one embedded CBE platform, is described. Section 7 concludes this paper.

2 Preliminaries

In the proposed system, boosting-based detection and histogram-based tracking with a particle filter are adopted for the detection and tracking phases, respectively. In this section, an overview of boosting and histogram-based tracking is described.

2.1 Boosting

Boosting is one ensemble learning method with which an accurate classifier is constructed by combining weak hypotheses learned by a weak learning algorithm. The obtained classifier consists of weak hypotheses and a combiner, and output is computed by the weighted votes of weak hypotheses. In the proposed scheme, AdaBoost [12], one of the most popular methods based on boosting, is adopted for construction of an accurate classifier. AdaBoost's learning flow is shown as follows.

Algorithm 2.1: AdaBoost$(h, H, (x_1, y_1), \dots, (x_n, y_n), m, l, T)$

```
for $i \leftarrow 1$ to $n$
  do if $y_i == 1$
  then $w_{1,i} = \frac{1}{2m}$
  else $w_{1,i} = \frac{1}{2l}$
for $t \leftarrow 1$ to $T$
  do
    for $i \leftarrow 1$ to $n$
      do $w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
    for $j \leftarrow 1$ to $H$
      do $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$
    Choose classifier $h_t$with the lowest error $\epsilon_t$
    for $i \leftarrow 1$ to $n$
      do $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}, \beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
    where $e_i = 0$ if example $x_i$ is classified correctly, $e_i = 1$ otherwise
Final strong classifier is: $h(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \alpha_t = \log 1/\beta_t \\ 0 & \text{otherwise} \end{cases}$
```

where x is an input sample and y indicates a label of the sample. Input is a negative sample if $y = 0$, and input is a positive sample if $y = 1$. T is the number of classifiers which strong classifier consists of, m and l are the number of negative and positive examples, respectively, h is a set of weak classifiers, and H is the number of sets of weak classifiers.

2.2 Histogram-Based Tracking

Histogram-based tracking is a particle filter-based tracking scheme in which state space, the state transition, and how to compute likelihood must be defined. In the rest of this subsection, state space, state transition, and a computation method of likelihood used in histogram-based tracking are described.

State Space. In the histogram-based tracking scheme, each particle of the distribution represents an rectangle and is given as:

$$\boldsymbol{s}_t = \{x_t, y_t, x_{t-1}, y_{t-1}, w_0, h_0, a_t, a_{t-1}\}, \tag{1}$$

where x_t and y_t specify the current location of rectangle, x_{t-1} and y_{t-1} the previous location, w_0 and h_0 specify the initial width and height of the rectangle, and a_t and a_{t-1} specify the scale change corresponding to the initial width and height.

State Transition. In histogram-based tracking, the probability distribution of a tracking target at the next time step is represented by:

$$q_B^*(\mathbf{x}_t|\mathbf{x}_{0:t-1}, Y_{1:t}) = \alpha q_{ada}(\mathbf{x}_t|\mathbf{x}_{t-1}, Y_t) + (1-\alpha)p(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{2}$$

where $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ shows the distribution of the previous time step and q_{ada} is the probability distribution derived from the detection results.

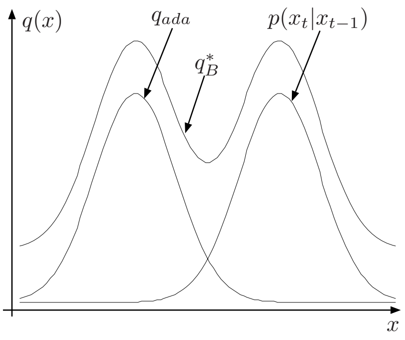

Fig. 1. State Transition

Figure 1 shows state transition by the above equation.

In this scheme, detection results are used for the state transition to enhance tracking accuracy, as shown in the above figure.

Likelihood Computation. In this scheme, likelihood is computed by using HSV histogram [13] as follows.

First, ξ, which is the Bhattacharyya distance between K^*, the HSV histogram of the area detected by the learning machine constructed by boosting, and $K({s'}_t^{(i)})$, which is the HSV histogram of predicted sample ${s'}_t^{(i)}$, are calculated by:

$$\xi[K^*, K({s'}_t^{(i)})] = \left[1 - \sum_{n=1}^{M} \sqrt{k^*(n)k(n;{s'}_t^{(i)})}\right]^{\frac{1}{2}}, \tag{3}$$

where $k^*(n)$ and $k(n;{s'}_t^{(i)})$ are the elements of K^*,$K({s'}_t^{(i)})$, respectively, and M means the size of the histogram.

Next, likelihood $\pi_t^{(i)}$ of sample ${s'}_t^{(i)}$ is computed by:

$$\pi_t^{(i)} = \exp\left(-\lambda\xi^2[K^*, K({s'}_t^{(i)})]\right), \tag{4}$$

where λ is a constant defined experimentally based on its application.

3 Overview of Cell Boradband Engine

In this section, Cell Broadband Engine architecture is summarized and parallel programming on CBE is introduced.

3.1 Architecture

Cell Broadband Engine (Cell) is a multi-core processor jointly developed by SONY, Toshiba, and IBM. Fig. 2 shows its architecture. A Cell is composed of one "Power Processor Element" (PPE) and eight "Synergistic Processor Elements" (SPE). PPE is the Power Architecture-based core that handles most of the computational workload, and SPE is a RISC processor with 128-bit SIMD organization for stream processing. PPE and SPEs are linked by an internal high speed bus called "Element Interconnect Bus" (EIB).

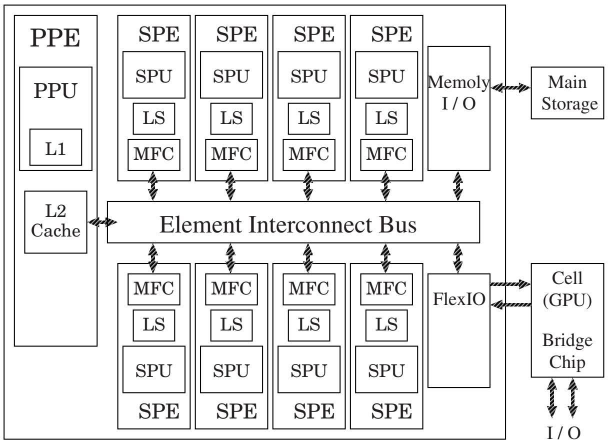

Fig. 2. Cell Broadband Enginearchitecture

PPE works with conventional operating systems due to its similarity to other 64-bit PowerPC processors. It also acts as the controller for multiple SPEs. Each SPE can operate independently when PPE boots up the SPE.

With current Cell generation, each SPE contains a 256 KB instruction and data local memory area called "Local Store," which does not operate as a conventional CPU cache. Then a programmer explicitly writes DMA operation code to transfer data between the main memory and the local store. SPE contains 128×128 register file. This feature enables the SPE compiler to optimize memory access to explore instruction level parallelism.

3.2 Parallel Implementation

We optimize our object recognition system for the Cell to realize real-time processing. This section shows the Cell specific programming methods which are suitable for CBE architecture.

- Multiple SPEs
  First, separate the processing into several groups so that multiple SPEs independently operate each processing group. Examples of image processing include filter processing with 4 SPEs; divide the image into 4 blocks and allocate one block to one SPE. Note that instruction and data local memory area in SPE must be less than 256 KB.

- Single Instruction Multiple Data (SIMD)
  An SPE contains 128-bit SIMD units and can operate on 16 8-bit integers, eight 16-bit integers, four 32-bit integers, or four single precision floating-point numbers in a single clock cycle.
- Loop unrolling
  An SPE contains a 128×128 register file. Unrolling loops increase register usage in a single iteration, but decrease the number of memory accesses and loop overhead instructions (branch or increment instructions).

4 Parallel Implementation

In this section, parallel implementation of boosting-based detection and histogram-based tracking on a Cell Broadband Engine are described.

4.1 Boosting-Based Detection

An object detection scheme based on boosting with haar-like features is executed as follows:

1) generate integral image of an input image,
2) search objects by scanning the whole input image with a constructed detector,
3) enlarge the scale of features used in the detector,
4) terminate detection if the size of features becomes greater than the size of an input image, or else go to 2.

The detection scheme can be performed by scaling an input image instead of scaling features as follows:

1) generate an integral image of an input image,
2) search objects by scanning the entire input image with a constructed detector,
3) scale down an input image,
4) terminate detection if an input image becomes smaller than the features, or else go to 1.

The latter scheme requires computational cost for generating shrunk images, but it is suitable for parallel implementation by specialized hardware or SIMD processor because the feature size is fixed. Furthermore, the authors showed that the latter scheme can achieve identical accuracy as the former scheme. Therefore, we adopt the latter scheme, which is expected to be suitable for the SIMD operation of SPE.

Integral images used for the detection phase are generated by :

$$I(x,y) = \sum_{n=0}^{y} \sum_{m=0}^{x} f(m,n), \tag{5}$$

where $f(m,n)$ and $I(x,y)$ are the luminance of image f at (m,n) and an integral image, respectively. Using the integral image, we obtain S_{ABCD}, which is the sum of the luminance of the area enclosed by points A, B, C, and D shown as Fig. 3, by:

$$S_{ABCD} = I(D) - I(C) - I(B) + I(A). \tag{6}$$

This operation includes only four load operations and three arithmetic operations.

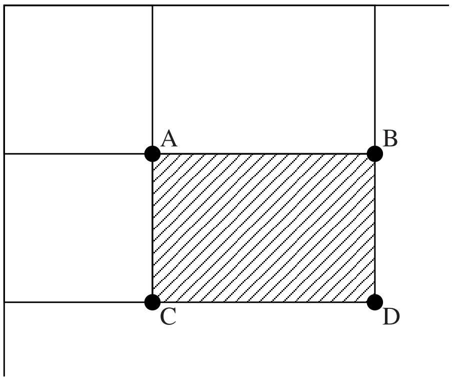

Fig. 3. Computation using integral images

In this implementation, the generation and the scaling of integral images are performed by PPE, and their detection using features is operated on SPEs. Here, each detection, which corresponds to different scales, is individually mapped to each SPE. By this partition of the processing of the detection phase, applying features, the generation of integral images, and the scaling of integral images are executed in parallel, which reduces the total processing time.

In each SPE, detection by features is computed in parallel by applying the SIMD operation. Detection is performed by moving the detection window to the

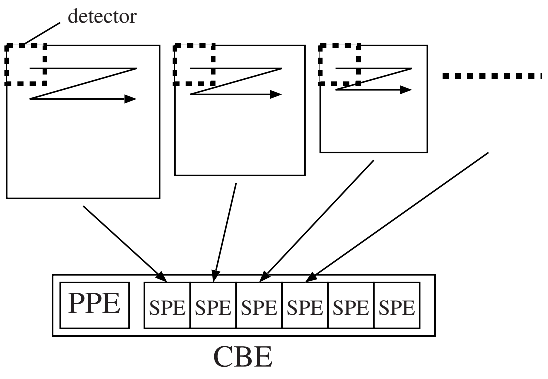

Fig. 4. Parallel execution of detection by multiple SPEs

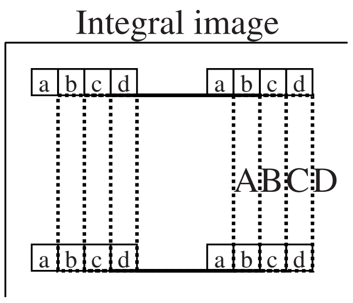

Fig. 5. Parallel computation of sum of luminance by SIMD operation

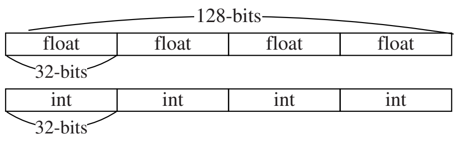

Fig. 6. SIMD vector

adjacent coordinates. In this phase, four detection operations can be executed in parallel because the SIMD vector of SPE can simultaneously operate four int variables, as shown in Fig. 6. By this parallel operation, four sums corresponding to A, B, C, and D are obtained, as shown in Fig. 5.

4.2 Histogram-Based Tracking

In an object tracking scheme based on particle filters, the probability distribution of the tracking target is represented by the density of particles. A particle filter consists of the following three steps: state transition, likelihood estimation, and resampling. Generally, likelihood estimation requires the most computational cost in these operations, and state transition and likelihood estimation can be operated in parallel because there is no dependence between each particle. Resampling cannot be executed in parallel; however it requires less computational power, so we use PPE for resampling in this implementation.

Applying SIMD operations to histogram calculation, which requires the most computational power in the computation of likelihood, is difficult because it consists of memory accesses to Lookup and histogram tables. Therefore, we apply the SIMD operation to the computation of Bhattacharyya distance, which requires the second most computational power. Applying the SIMD operation to the computation of Bhattacharyya distance is easy because it consists of an operation to N array elements. Since this computation requires normalization of the histogram, this process is also implemented with the SIMD operation.

Here, it is necessary for the computation of likelihood to access HSV images. However, storing whole HSV images in the local store, which only SPE can directly access, is difficult because its size is limited to 256 KBytes.

5 Real-Time Object Recognition by Combining Detection and Tracking

In the previous section, the parallel implementation of detection and tracking on Cell were described. To realize a real-time object recognition system by combining these processes, allocating SPEs for them that consider required computational power is important. In this section, first, we discuss load balance for object recognition and then introduce a real-time object recognition system based on SONY$^{®}$ Playstation 3 [14], one of the most widely used Cell platform.

5.1 Load Balance on Cell for Object Recognition

The relation between processing time and the number of SPEs for detection and tracking is measured for optimal load balance on the Cell. In this experiment, the size of the detection and tracking images is 320×240, the size of the features is 24×24, the number of particles is 128 for a tracking target. Input image size is started from 320×240 and ended up 32×24, and the size is scaled down 83 percent at each iteration. These parameters are decided experimentally to achieve both real-time processing and high recognition performance. The results are shown in Table 1.

Table 1. Processing time of detection and tracking

Number of SPEs	1	2	3	4	5
detect objects	73.47	51.40	45.06	46.04	48.46
track objects	27.44	13.55	12.11	7.32	12.69

The processing time of detection decreases as the number of SPEs increases; however, the time increases if the number of SPEs becomes greater than three. The processing time of tracking decreases as the number of SPEs increases; however, the time increases if the number of SPEs becomes greater than five because the required time to manage SPEs sometimes becomes greater.

In this system, the number of available SPEs is six, because Playstation 3 is adopted as a Cell platform. Considering processing time, we should allocate two or three SPEs to the detection process. If two SPEs are allocated to detection, about eight targets can be tracked while the detection process for the next frame is performed. In this implementation, we use OpenCV on Cell[15] package, which the authors cooperate with members of OpenCV on the Cell project to develop, for detection and we adopt a software cache implemented in Cell/BE Software Development Kit[16] to deal with entire input image on each SPE's local store for tracking.

In this case, the object recognition performance achieves 18 fps. If three SPEs are allocated to detection, about three targets can be tracked while the detection process for the next frame is performed. In this case, the object recognition performance achieves 22 fps.

5.2 Real-Time Implementation on Playstation 3

Based on the above results, we constructed a real-time object recognition system using Playstation 3 and Qcam Orbit MP QVR-13R, one USB camera. The

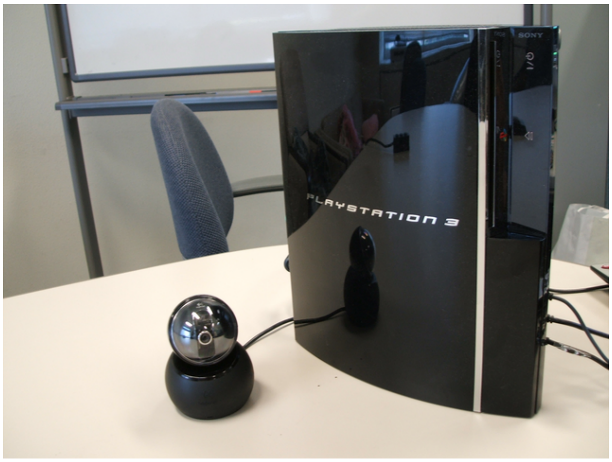

Fig. 7. Real-time object recognition system

following operations are required in addition to detection and tracking when a USB camera is used for real-time processing:

1) acquire images from a USB camera (640x480 pixels, RGB image)
2) shrink input images to 320×240 and convert color to grayscale and HSV images.

Table 2 shows the required processing time for the above operations.

Table 2. Processing time of miscellaneous functions

Number of SPEs	1	2	3	4	5
retrieve frame	130.38	129.47	129.61	130.03	131.60
convert color and resize	18.97	17.93	17.79	18.73	17.87

By this result, this system achieves real-time object detection at about 7 fps. In this implementation, image aquisition from USB camera becomes dominant.

6 Demonstration

Figure 8 shows the face recognition results with the proposed system. In these figures, white and green rectangles correspond to detected and tracked objects, respectively. In the 60th frame, both white and green rectangles are shown around the target face because both detection and tracking succeed. In the 103rd and the 248th frame, detection fails but the position of the face is indicated by the tracking result. In the 263rd frame, the face is both successfully detected and tracked.

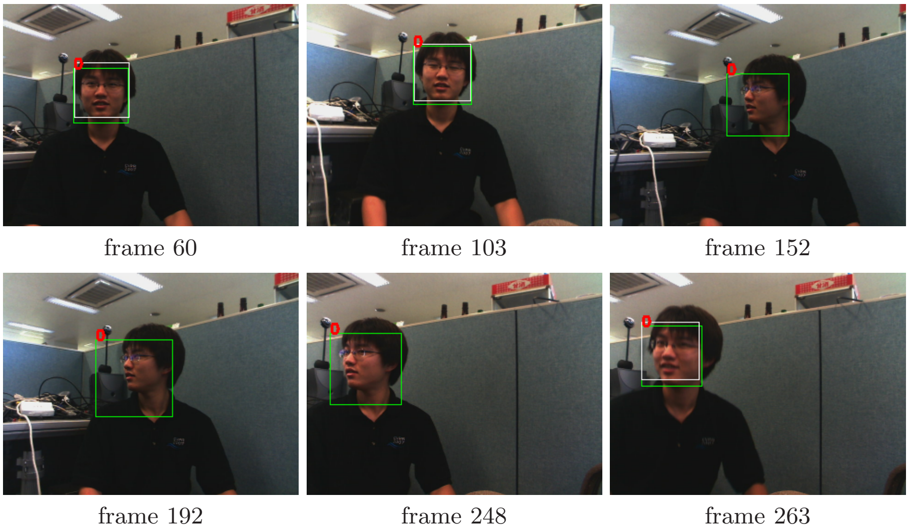

Fig. 8. Face recognition result

7 Conclusion

In this paper, we showed the parallel implementation of boosting-based detection and histogram-based tracking on Cell, discussed load balance on Cell for object recognition, and showed sample implementation of a real-time object recognition system based on Playstation 3. We showed that Cell can ideally achieve real-time object recognition on QVGA video at 22 fps for three targets and 18 fps for eight targets. Furthermore, real-time face detection is demonstrated with a real-time object recognition system implemented on SONY® Playstation 3, one of the most widely used Cell platforms.

In the future, we will improve the image acquisition performance from the USB camera to reveal Cell performance with the widely used Playstation 3.

References

1. Hu, W., Tan, T., Wang, L., Maybank, S.: A survey on visual surveillance of object motion and behaviors. IEEE Trans. on SMC 34, 334–352 (2004)
2. Zhao, L., Thorpe, C.E.: Stereo- and neural network-based pedestrian detection. IEEE Trans. on ITS 01, 148–154 (2000)
3. Oren, M., Papageorgiou, C., Sinha, P., Osuna, E., Proggio, T.: Pedestrian detection using wavelet templates. In: Proc. of CVPR, pp. 193–199 (1997)
4. Papageorgiou, C., Poggio, T.: Trainable pedestrian detection. In: Proc. of ICIP, vol. 4, pp. 35–39 (1999)
5. Viola, P., Jones, M.J., Snow, D.: Detecting pedestrians using patterns of motion and appearance. International Journal of Computer Vision 63, 153–161 (2005)
6. Soga, M., Kato, T., Ohta, M., Ninomiya, Y.: Pedestrian detection using stereo vision and tracking. In: Proc. of The 11th World Congress on Intelligent Transport Systems (2004)
7. Ashida, J., Miyamoto, R., Tsutsui, H., Onoye, T., Nakamura, Y.: Probabilistic pedestrian tracking based on a skeleton model. In: Proc. of ICIP, pp. 2825–2828 (2006)
8. Miyamoto, R., Sugano, H., Saito, H., Tsutsui, H., Ochi, H., Hatanaka, K., Nakamura, Y.: Pedestrian recognition in far-infrared images by combining boosting-based detection and skeleton-based stochastic tracking. In: Proc. of PSIVT, pp. 483–494 (2006)
9. Masayuki, H., Nakahara, K., Sugano, H., Nakamura, Y., Miyamoto, R.: A specialized processor suitable for adaboost-based detection with haar-like features. In: Proc. of CVPR (2007)
10. Brown, M.Z., Burschka, D., Hager, G.: Advances in computational stereo. IEEE Trans. on PAMI 25, 993–1008 (2003)
11. Okuma, K., Taleghani, A., de Freitas, N., Little, J.J., Lowe, D.G.: A boosted particle filter: Multitarget detection and tracking. In: Proc. of ECCV, pp. 28–39 (2004)
12. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
13. Perez, P., Hue, C., Vermaak, J., Gangnet, M.: Color-based probabilistic tracking. In: Proc. of ECCV (2002)
14. Playstation3 (2007), `http://www.us.playstation.com/PS3`
15. OpenCV on The Cell (2007), `http://cell.fixstars.com/opencv/index.php/OpenCV_on_the_Cell`
16. Cell/BE software development kit (SDK) version 2.1 (2007), `http://www.bsc.es/plantillaH.php?cat_id=301`